Refine Search

Search Results -

Terms	Documents
L12 and (((last or later or first) near6 (replac$6 or substitut$3 or remap$4)) with (lock$4 or unlock4))	0

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L13

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, August 23, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L13	L12 and (((last or later or first) near6 (replac$6 or substitut$3 or remap$4)) with (lock$4 or unlock4))	0	L13
L12	L10 and ((last or later or first) near6 (replac$6 or substitut$3 or remap$4))	4	L12
L11	L10 and (way near6 hit)	0	L11
L10	L9 and ((order$4 or arrang$6 or position$4 or sequen$4) with (replac$6 or substitut$3 or remap$4))	10	L10
L9	L8 and (way near6 (compar$6 or select$4))	44	L9
L8	cach$3 and ((way or partition or section) near6 (replac$6 or substitut$3 or remap$4))	270	L8
DB=PGPB,USPT; PLUR=YES; OP=ADJ			
L7	L5 and (((last or later or first) near6 (replac$6 or substitut$3 or remap$4)) with (lock$4 or unlock4))	2	L7
L6	L5 and ((way number) near6 (compar$6 or select$4))	2	L6
L5	L4 and ((last or later or first) near6 (replac$6 or substitut$3 or remap$4))	72	L5
L4	L3 and (way near6 hit)	103	L4
L3	L2 and ((order$4 or arrang$6 or position$4 or sequen$4) with (replac$6 or substitut$3 or remap$4))	198	L3

L2	L1 and (way near6 (compar$6 or select$4))	460	L2
L1	cach$3 and ((way or partition or section) near6 (replac$6 or substitut$3 or remap$4))	1719	L1

END OF SEARCH HISTORY

Your search matched **0** of **1062489** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

cach$ and (way <near/4> replac$) and ((last or later or Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

Results:
No documents matched your query.

Terms used
cach and **way** **near/4** **replac** and **last** or **later** or **first** **near/6** **replac** or **substitut** or **remap** and **way** **number** **near/6** **compa**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search
Try this search in The ACM Gu

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevan

1 Parallel execution of prolog programs: a survey

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo

July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 4

Full text available: pdf(1.95 MB) Additional Information: full citation, abstract, references, citings, index terms

Since the early days of logic programming, researchers in the field realized the potential for exploitation of present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and th referential transparency, among other characteristics, make logic programs interesting candidates for obtai speedups through parallel execution. At the same time, the fact that the typical applications of logic progra frequently involve irregular computatio ...

Keywords: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

2 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collabora research**

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time d often used to obtain a better understanding of the execution of the application. The visualization tool we us event tracer developed at the University of Waterloo. However, these diagrams are often very complex and provide the user with the desired overview of the application. In our experience, such tools display repeated of non-trivial commun ...

3 Query evaluation techniques for large databases

Goetz Graefe

June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2

Full text available: pdf(9.37 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for manipulating large sets and sequences will be required to provide acceptable performance. The advent of o oriented and extensible database systems will not solve this problem. On the contrary, modern data models the problem: In order to manipulate large sets of complex objects as efficiently as today's database system simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems object-oriented database systems, operator model of parallelization, parallel algorithms, relational database set-matching algorithms, sort-hash duality

4 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

In the last three decades a large number of compiler transformations for optimizing programs have been im Most optimizations for uniprocessors reduce the number of instructions executed by the program using tran based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-perf superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, supersca processors, vectorization

5 Conception, evolution, and application of functional programming languages
Paul Hudak
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: pdf(5.19 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

The foundations of functional programming languages are examined from both historical and technical pers Their evolution is traced through several critical periods: early work on lambda calculus and combinatory ca Iswim, FP, ML, and modern functional languages such as Miranda[1] and Haskell. The fundamental premises o functional programming methodology stands are critically analyzed with respect to philosophical, theoretica pragmatic concerns. ...

6 The CLP(R) language and system
Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, Roland H. C. Yap
May 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 14 Issue 3

Full text available: pdf(3.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The CLP R programming language is defined, its underlying philosophy and programming methodology are important implementation issues are explored in detail, and finally, a prototype interpreter is described. CLP designed to be an instance of the Constraint Logic Programming Scheme ...

Keywords: constraints, logic programming

7 The family of concurrent logic programming languages
Ehud Shapiro
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that range of both known and novel concurrent programming techniques. Being logic programming languages, t many advantages of the abstract logic programming model, including the logical reading of programs and c the convenience of representing data structures with logical terms and manipulating them using unification, amenability to metaprogrammin ...

8 Parallelizing nonnumerical code with selective scheduling and software pipelining
Soo-Mook Moon, Kemal Ebcioğlu
November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6

Full text available: pdf(543.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Instruction-level parallelism (ILP) in nonnumerical code is regarded as scarce and hard to exploit due to its In this article, we introduce a new code-scheduling technique for irregular ILP called "selective scheduling" used as a component for superscalar and VLIW compilers. Selective scheduling can compute a wide set of i operations across all execution paths based on renaming and forward-substitution and can compute availab

Keywords: VLIW, global instruction scheduling, instruction-level parallelism, software pipelining, speculativ motion, superscalar

9 Launching the new era
Kazuhiro Fuchi, Robert Kowalski, Koichi Furukawa, Kazunori Ueda, Ken Kahn, Takashi Chikayama, Evan Tick
March 1993 **Communications of the ACM**, Volume 36 Issue 3
Full text available: pdf(3.45 MB) Additional Information: full citation, references, index terms, review

10 Continuous program optimization: A case study
Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4
Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced b engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these pro delay code generation until load time. This is the earliest point at which a piece of software can be fine-tune actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

11 A structured approach for the definition of the semantics of active databases
Piero Fraternali, Letizia Tanca
December 1995 **ACM Transactions on Database Systems (TODS)**, Volume 20 Issue 4
Full text available: pdf(4.15 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

Active DBMSs couple database technology with rule-based programming to achieve the capability of reactio database (and possibly external) stimuli, called events. The reactive capabilities of active databases are use spectrum of applications, including security, view materialization, integrity checking and enforcement, or he database integration, which makes this technology very promising for the near future. An active database s consists of ...

Keywords: active database systems, database rule processing, events, fixpoint semantics, rules, semantic

12 Distributed file systems: concepts and examples
Eliezer Levy, Abraham Silberschatz
December 1990 **ACM Computing Surveys (CSUR)**, Volume 22 Issue 4
Full text available: pdf(5.33 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

The purpose of a distributed file system (DFS) is to allow users of physically distributed computers to share storage resources by using a common file system. A typical configuration for a DFS is a collection of workst mainframes connected by a local area network (LAN). A DFS is implemented as part of the operating system the connected computers. This paper establishes a viewpoint that emphasizes the dispersed structure and decentralization of both data and con ...

13 Data caching issues in an information retrieval system
Rafael Alonso, Daniel Barbara, Hector Garcia-Molina
September 1990 **ACM Transactions on Database Systems (TODS)**, Volume 15 Issue 3
Full text available: pdf(2.11 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

Currently, a variety of information retrieval systems are available to potential users.... While in many cases systems are accessed from personal computers, typically no advantage is taken of the computing resources machines (such as local processing and storage). In this paper we explore the possibility of using the user's capabilities to cache data at the user's site. This would improve the response time of user queries albeit at incurring t ...

Keywords: cache coherency, data sharing, information retrieval systems

14 Design of a high-performance ATM firewall

Jun Xu, Mukesh Singhal

August 1999 **ACM Transactions on Information and System Security (TISSEC)**, Volume 2 Issue 3

Full text available: pdf(143.19 KB) Additional Information: full citation, abstract, references, index terms

A router-based packet-filtering firewall is an effective way of protecting an enterprise network from unautho However, it will not work efficiently in an ATM network because it requires the termination of end-to-end AT connections at a packet-filtering router, which incurs huge overhead of SAR (Segmentation and Reassembly approaches to this problem have been proposed in the literature, and none is completely satisfactory. In th present the hardware desig ...

Keywords: TCP/IP, asynchronous transfer mode, firewall, packet filtering, switch architecture

15 System-level power optimization: techniques and tools

Luca Benini, Giovanni de Micheli

April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 5 Issue 2

Full text available: pdf(385.22 KB) Additional Information: full citation, abstract, references, citings, index terms

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic sytem of a hardware platform and software layers. We consider the three major constituents of hardware that con energy, namely computation, communication, and storage units, and we review methods of reducing their e consumption. We also study models for analyzing the energy cost of software, and methods for energy-effic design and compilation. This survery ...

16 Design, implementation and testing of extended and mixed precision BLAS

Xiaoye S. Li, James W. Demmel, David H. Bailey, Greg Henry, Yozo Hida, Jimmy Iskandar, William Kahan, Suh Anil Kapur, Michael C. Martin, Brandon J. Thompson, Teresa Tung, Daniel J. Yoo

June 2002 **ACM Transactions on Mathematical Software (TOMS)**, Volume 28 Issue 2

Full text available: pdf(456.84 KB) Additional Information: full citation, abstract, references, citings, index terms, revi

This article describes the design rationale, a C implementation, and conformance testing of a subset of the Standard for the BLAS (Basic Linear Algebra Subroutines): Extended and Mixed Precision BLAS. Permitting internal precision and mixed input/output types and precisions allows us to implement some algorithms tha more accurate, and sometimes faster than possible without these features. The new BLAS are challenging t and test because there are many more subr ...

Keywords: BLAS, double-double arithmetic, extended and mixed precision

17 Experimental evaluation of a generic abstract interpretation algorithm for PROLOG

Baudouin Le Charlier, Pascal Van Hentenryck

January 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 1

Full text available: pdf(4.18 MB) Additional Information: full citation, abstract, references, citings, index terms

Abstract interpretation of PROLOG programs has attracted many researchers in recent years, partly becaus potential for optimization in PROLOG compilers and partly because of the declarative nature of logic program languages that make them more amenable to optimization than procedural languages. Most of the work, ho remained at the theoretical level, focusing on the developments of frameworks and the definition of abstrac This paper reports our effo ...

Keywords: PROLOG, abstract interpretation, fixpoint algorithm

18 4.2BSD and 4.3BSD as examples of the UNIX system

John S. Quarterman, Abraham Silberschatz, James L. Peterson

December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4

Full text available: pdf(4.07 MB) Additional Information: full citation, abstract, references, citings, index terms, revi

This paper presents an in-depth examination of the 4.2 Berkeley Software Distribution, Virtual VAX-11 Vers

(4.2BSD), which is a version of the UNIX Time-Sharing System. There are notes throughout on 4.3BSD, the system from the University of California at Berkeley. We trace the historical development of the UNIX syste conception in 1969 until today, and describe the design principles that have guided this development. We th the internal data structures and ...

19 Using certes to infer client response time at the web server

David Olshefski, Jason Nieh, Dakshi Agrawal

February 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 1

Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, index terms

As businesses continue to grow their World Wide Web presence, it is becoming increasingly vital for them to quantitative measures of the mean client perceived response times of their web services. We present Certe Response Time Estimated by the Server), an online server-based mechanism that allows web servers to est client perceived response time, as if measured at the client. Certes is based on a model of TCP that quantifi that connection drops have on mean ...

Keywords: Web server, client perceived response time

20 Hints for computer system design

Butler W. Lampson

October 1983 **ACM SIGOPS Operating Systems Review , Proceedings of the ninth ACM symposium on systems principles**, Volume 17 Issue 5

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Experience with the design and implementation of a number of computer systems, and study of many othe has led to some general hints for system design which are described here. They are illustrated by a number examples, ranging from hardware such as the Alto and the Dorado to applications programs such as Bravo